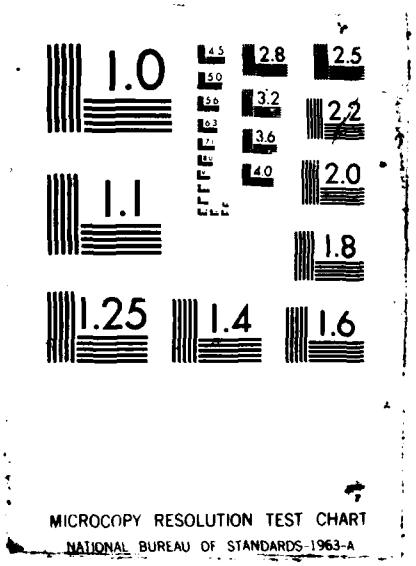
AD-A185 647 COMPUTING BLOCK-ANGULAR KARMARKER PROJECTIONS WITH APPLICATIONS TO STOCHA..(U) MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERATIONS ENG.. 1/1

UNCLASSIFIED J R BIRGE ET AL. DEC 86 TR-85-12-REV F/G 12/4 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

Computing Block-Angular Karmarkar Projections
with Applications to Stochastic Programming

John R. Birge
Dept. of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109, USA

Liqun Qi
Dept. of Applied Mathematics
Tsinghua University
Beijing, China
and
Dept. of Mathematics and Statistics
University of Pittsburgh
Pittsburgh, PA 15213, USA

Department of Industrial and Operations Engineering

The University of Michigan
College of Engineering
Ann Arbor, Michigan 48109-2117

(12)

Computing Block-Angular Karmarkar Projections with Applications to Stochastic Programming

John R. Birge
Dept. of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109, USA

Liqun Qi
Dept. of Applied Mathematics
Tsinghua University
Beijing, China
and
Dept. of Mathematics and Statistics
University of Pittsburgh
Pittsburgh, PA 15213, USA

Technical Report 85-12

Revised December 1986

Computing Block-Angular Karmarkar Projections with Applications to Stochastic Programming

John R. Birge* and Liqun Qi **

Abstract: We present a variant of Karmarkar's algorithm for block angular structured linear programs, such as stochastic linear programs. By computing the projection efficiently, we give a worst case bound on the order of the running time that can be an order of magnitude better than that of Karmarkar's standard algorithm. A related variant is applied to the dual program, and its implications for very large-scale problems are given.

* Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109, USA. His work was supported by National Science Foundation Grant NO. ECS-8304065, Office of Naval Research Grant N0014-86-k-0628, and Dalhousie University during a visit in the Department of Mathematics, Statistics, and Computing Science.

** Department of Applied Mathematics, Tsinghua University, Beijing, China, and Department of Mathematics and Statistics, University of Pittsburgh, PA 15213, U.S.A. This author's work was supported by the 1984-1985 Andrew Mellon Postdoctoral Fellowship at University of Pittsburgh.

1. Introduction

Block angular and dual block angular linear programs arise in a variety of applications, in particular, stochastic linear programming. Dual block angular linear programs (see, e.g., [10],[21],[29]) have the form:

$$\begin{array}{lllll} \min & c_0^T x_0 & +\sum_{k=1}^N c_k^T x_k & & \\ \text{subject to} & A_0 x_0 & & = b_0, & \\ & A_k x_0 & +W_k x_k & = b_k, k = 1, \ldots, N, & \\ & & x_k \geq 0, & k = 0, \ldots, N, & \end{array} \tag{1.1}$$

where the sizes of the matrices are consistent with $x_k \in \Re^{n_k}, k = 0, \ldots, N$, $b_k \in \Re^{m_k}, k = 0, \ldots, N$, where $m_k \leq n_k, k = 0, \ldots, N$, and we assume A_0 and W_k have full row ranks. We also assume that the blocks are large enough that $n_0 \leq \sum_{k=1}^N n_k$.

Stochastic linear programs with fixed recourse (SLPF) with discrete random elements have the form in (1.1). They can be formulated as:

$$\begin{array}{lllll} \min & c^T x & +Q(x) & & \\ \text{subject to} & Ax & & = b & \\ & & x \geq 0, & & \end{array} \tag{1.2}$$

where

$$Q(x) = \sum_{k=1}^N p_k Q(x, \xi^k)$$

and for each $k = 1, ..., N$, the recourse cost $Q(x, \xi^k)$ is obtained by solving the recourse problem:

$$Q(x, \xi^k) = \inf\{q^k y \mid Wy = h^k - T^k x, y \in \Re_+^{n_1}\},$$

$$\xi^k = (q^k, h^k, T^k),$$

$$p_k = \text{prob}[\xi(\omega) = \xi^k].$$

Substituting the expressions for Q in (1.2) , we obtain a problem in the form of (1.1) with $W = W_k$, $T^k = \bar{a}_k$, and $p_k q^k = c_k$, for $k = 1, \ldots, N$. This problem has $n = n_0 + Nn_1$ variables and $m = m_0 + Nm_1$ constraints. The methods for solving it include: the L-shaped method, proposed by Van Slyke and Wets [25]; the decomposition method, proposed by Dantsig and Madansky [11]; and the basis factorization method, proposed by Strasicky [23], and modified by Kall [17] and Wets [27]. The first method directly solves SLPF in form (1.1), while the other two solve the dual. Birge [3][4] discusses the relationship among them. Also see [27][28] for other references.

If $T_1 = \cdots = T_N$, then (1.1) has a staircase structure. This type of problem is widely encountered in the context of dynamical systems, i.e., discrete versions of continuous linear programs or linear control problems [7][8][13][21][25][28].

In this paper, we study the approach of Karmarkar's projective algorithm for linear programming [19] to solve (1.1). We show that the worst-case performance bound can be improved for this special structure by reducing the complexity of computing the projections. For large ($N \sim n_0 \sim n_1$), this improvement is an order of magnitude. We also show that refinements of a special case of (1.1) can be solved in at most a number of iterations which is linear in the *additional* data. These results indicate the potential of interior point methods in stochastic linear programming, but extensive computational studies will be required to determine their practical average-case behavior.

A standard form version of Karmarkar's algorithm is briefly described in Section 2. For L, the length of the input, this method requires $O(nL)$ iterations. The order of each iteration is dominated by the projection operation requiring generally $O(n^3L)$ operations. In Section 3, by means of the Sherman-Morrison-Woodbury formula and block matrix method, we explore the advantage of the special structure of (1.1) and obtain a method for the projection with order $O(N(n_1^3 + n_0^2 n_1 + n_0 n_1^2)) = O(n^2 L)$ if $n_0 \sim n_1 \sim N$. A further reduction is possible using rank-one updates of an inverse or Cholesky factorisation. Section 4 presents another variant of Karmarkar's algorithm to solve the dual of (1.1) (with block angular structure). Section 5 describes the implementation of the variants in sequential approximation methods and shows how worst case bounds can be improved in these procedures. Section 6 provides some special cases where additional computational savings are possible.

2. Karmarkar's projective algorithm

We briefly describe a standard form versionof Karmarkar's projective algorithm as in Gay[14], Ye[30], or Lustig[20] (with known objective value). We choose the standard form because we believe it is more practical for computation than various canonical forms. We also assume an unknown objective value and use Todd and Burrell's [24] method for updating a lower bound on the objective value. We use an initial lower bound as is often available in practice. An alternative is Anstreicher's [1] method to obtain an initial lower bound.

Suppose a linear program:

$$\begin{array}{ll} \min & c^T x \\ \text{subject to} & Ax = b, \\ & x \geq 0. \end{array} \tag{2.1}$$

where $x \in \mathbb{R}^n$, $c \in Z^n$, $b \in Z^m$, $A \in Z^{m \times n}$ with optimal value $c^T x^* = z^*$.

Suppose that we have a strictly interior feasible point $\bar{a}$ of (2.1), i.e.,

$$A\bar{a} = b, \bar{a} > 0, \tag{2.2}$$

a lower bound l^0 on z^*, and that the set of feasible solutions in (2.1) is bounded. Note that if we do not have a feasible solution, then we can solve a phase-one problem as in Karmarkar [19].

Karmarkar's algorithm creates a sequence of points $x^0, x^1, ..., x^k$ by these steps:

1. $x^0 = \bar{a}, k = 0$. A lower bound $l^0 \leq z^*$.

2. If $c^T x^k - l^k$ is small enough, i.e., less than a given positive number ϵ, then stop. Otherwise go to 3.

3. Let $D = \text{diag}\{x_1^k, \ldots, x_n^k\}$, $\hat{A} := [AD, -b]$, and let $P_{\hat{A}}$ be the projection onto the null space of $\hat{A}$. Find

$$u = P_{\hat{A}} \begin{pmatrix} Dc \\ 0 \end{pmatrix}$$

and

$$v = P_{\hat{A}} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.3}$$

and let $\mu(l^k) = \min\{u_i + l^k v_i : i = 1, \ldots, n+1\}$.

If $\mu(l^k) \leq 0$, let $l^{k+1} = l^k$.

Otherwise, let $l^{k+1} = \min\{u_i/v_i : v_i > 0, i = 1, \ldots, n+1\}$.

4. Let $c_p = u - l^{k+1} v - (c^T x^k - l^{k+1}) e/(n+1)$, where $e = (1, \ldots, 1)^T \in \mathbb{R}^{n+1}$.

Let

$$g' = \frac{1}{n+1} e - \frac{\alpha}{\sqrt{n(n+1)}} \frac{c_p}{\|c_p\|_2}.$$

Suppose $g' = \bar{g}/g_{n+1}$, where $\bar{g} \in \mathbb{R}^n$, $g_{n+1} \in \mathbb{R}$. Then $x^{k+1} = D\bar{g}/g_{n+1}$. $k = k+1$, go to 1.

The main computational effort at each step k is to compute the projections in (2.3). This effort is generally $O(n^3L)$. Since this algorithm converges in $O(nL)$ steps (see Ye[30]), the algorithm has complexity $O(n^4L^2)$. Karmarkar [19] uses a rank–one updating scheme to reduce the complexity to $O(n^{3.5}L^2)$. We discuss this approach and a related updating method of Shanno[22] in Section 3.

3. Computation Reduction

We consider (1.1) and assume $n_1 = n_2 = \cdots = n_N$ and $m_1 = m_2 = \cdots = m_N$ without loss of generality. If an element of a vector or a matrix needs to be specified, we use parentheses, e.g., $(A_k)_{ij}$.

Let $n = n_0 + Nn_1, m = m_0 + Nm_1$,

$$x = \begin{pmatrix} x_0 \\ \vdots \\ x_N \end{pmatrix}, c = \begin{pmatrix} c_0 \\ \vdots \\ c_N \end{pmatrix}, b = \begin{pmatrix} b_0 \\ \vdots \\ b_N \end{pmatrix},$$

and

$$A = \begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & W_1 & \cdots & 0 \\ \vdots & 0 & \ddots & 0 \\ A_N & 0 & \cdots & W_N \end{pmatrix}.$$

Then (1.1) can be expressed exactly the same as (2.1) for application of Karmarkar's algorithm as described in Section 2. Suppose $a = (a_0^T, \ldots, a_N^T)^T$ is the current iteration point of Karmarkar's algorithm. Let $D_k = \text{diag}\{(a_k)_1, \ldots, (a_k)_{n_k}\}$, $D = \text{diag}\{D_0, \ldots, D_N\}$. As discussed in Section 2, the main computational work at each step of Karmarkar's algorithm is to compute the projections in (2.3). The projection can be written as

$$P_{\hat{A}} = (I - \hat{A}^T(\hat{A}\hat{A}^T)^{-1}\hat{A}),$$

where $(\hat{A}\hat{A}^T) = AD^2A^T + bb^T := M$. In this case, the work is dominated by computing M^{-1}. Using an explicit inverse would generally not be the most efficient method for computing the projection but the complexity is the same in other approaches. In the following, we use the inverse notation to simplify our discussion of reduced complexity order for (1.1). We then describe how this may be used in practice to solve systems with M.

Theorem 3.1. Let $S_0 = -I_2 \in \Re^{m_0 \times m_0}$, $S_k = W_k D_k^2 W_k^T$, $k = 1, \ldots, N$, $S = \text{diag}\{S_0, \ldots, S_N\}$. Then $S^{-1} = \text{diag}\{S_0, S_1^{-1}, \ldots, S_N^{-1}\}$. Let $B_k = (A_k b_k)$ for $k = 0, \ldots, N, \overline{D}_0 = \text{diag}\{D_0, 1\}$, I_1 and I_2 be unit

matrices in $\Re^{n_0+1}$ and $\Re^{m_0}$ respectively. Let

$$G_1 = (\overline{D}_0)^{-2} + \sum_{k=0}^{N} B_k^T S_k^{-1} B_k, G_2 = -B_0 G_1^{-1} B_0^T, \tag{3.1}$$

$$U = \begin{pmatrix} B_0 & I_2 \\ B_1 & 0 \\ \vdots & \vdots \\ B_N & 0 \end{pmatrix}.$$

If G_1 is invertible, then G_2 and M are invertible and

$$M^{-1} = S^{-1} - S^{-1}U \begin{pmatrix} I_1 & G_1^{-1}B_0^T \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} I_1 & 0 \\ 0 & G_2^{-1} \end{pmatrix} \begin{pmatrix} I_1 & 0 \\ B_0 & I_2 \end{pmatrix} \begin{pmatrix} G_1^{-1} & 0 \\ 0 & I_2 \end{pmatrix} U^T S^{-1}.$$

Proof: In Karmarkar's algorithm, a is positive, i.e., D is always invertible. Since W_k has full row rank, S_k is invertible for $k = 1, \ldots, N$. Therefore, S is invertible. Let $\overline{D} = \text{diag}\{\overline{D}_0, I_2\}$. Let

$$G = \begin{pmatrix} G_1 & -B_0^T \\ -B_0 & 0 \end{pmatrix}, \tilde{U} = \begin{pmatrix} B_0\overline{D}_0 & I_2 \\ \vdots & \vdots \\ B_N\overline{D}_0 & 0 \end{pmatrix} = U\overline{D}.$$

It is seen that $M = S + \tilde{U}\tilde{U}^T$. According to the Sherman-Morrison-Woodbury formula [16], M is invertible and

$$\begin{aligned} M^{-1} &= S^{-1} - S^{-1}\tilde{U}(I + \tilde{U}^T S^{-1}\tilde{U})^{-1}\tilde{U}^T S^{-1} \\ &= S^{-1} - S^{-1}U(\overline{D}^{-2} + U^T S^{-1}U)^{-1}U^T S^{-1} \\ &= S^{-1} - S^{-1}UG^{-1}U^T S^{-1} \end{aligned} \tag{3.3}$$

if and only if $(I + \tilde{U}^T S^{-1}\tilde{U})$ is invertible, i.e., G is invertible.

Suppose G_1 is invertible. Since G_1 is a symmetric matrix, so is G_1^{-1}, and we can write $G_1^{-1} = G_1^{-1/2}G_1^{-1/2}$, where $G_1^{-1/2}$ is also a symmetric matrix (although possibly complex). Since A_0 has full row rank, $B_0G_1^{-1/2}$ also has full row rank. Since $n_0 + 1 \geq m_0$, $G_2 = -(B_0G_1^{-1/2})(B_0G_1^{-1/2})^T$ is also invertible. It is easy to see that

$$G \begin{pmatrix} I_1 & G_1^{-1}B_0^T \\ 0 & I_2 \end{pmatrix} \begin{pmatrix} I_1 & 0 \\ 0 & G_2^{-1} \end{pmatrix} \begin{pmatrix} I_1 & 0 \\ B_0 & I_2 \end{pmatrix} \begin{pmatrix} G_1^{-1} & 0 \\ 0 & I_2 \end{pmatrix} = \begin{pmatrix} I_1 & 0 \\ 0 & I_2 \end{pmatrix} = I_{m_0+n_0+1}.$$

Hence, G is invertible. According to (3.5), M is invertible and (3.4) holds.■

By using (3.2), we can now reduce the complexity of computing the projection $P_{\hat{A}}$.

Theorem 3.2 (Complexity). Suppose G_1 is invertible then the overall running time of computing projections (2.3) is $O(N(n_1^3 + n_1^2 n_0 + n_0^2 n_1)L)$.

Proof: At each step, we have for $\overline{n} = \max\{n_0, n_1\}$,

work	number of operations
calculate S^{-1}	$O(Nn_1^3)$
calculate G_1	$O(Nn_1^2 n_0 + n_1 n_0^2)$
calculate G_2	$O(n_0^3)$
calculate G_1^{-1} and G_2^{-1}	$O(n_o^3)$
multiply A with a vector	$O(m\overline{n})$
multiply S^{-1} with a vector	$O(mn_1)$
multiply U with a vector	$O(mn_0)$

Other efforts are small compared with the operations above. According to our assumptions, the number of arithmetic operations is dominated by $O(N(n_1^3 + n_1^2 n_0 + n_1 n_0^2))$. Since each arithmetic operation needs a precision of $O(L)$ we obtain our conclusion. ∎

As we stated above,in practice, we would not compute M^{-1} explicitly. The work in (2.3) is dominated by the effort to solve systems of the form

$$Mv = u, \tag{3.4}$$

using

$$v = p - r, \tag{3.5}$$

where

$$\begin{aligned} Sp &= u, \\ G_q &= U^T p, \\ Sr &= Uq. \end{aligned} \tag{3.6}$$

The systems in (3.6) require solving systems with S_k,computation of G_1 and G_2 and solving systems with G_1 and G_2. In practice, we would find a Cholesky factorisation of each S_k, use them to find G_1 and G_2 and find Cholesky factorisations of G_1 and G_2.

Karmarkar [19] used a rank–one updating technique for solving a canonical form linear program. In this case, we let

$$\delta = \frac{1}{n}\sum_{j=1}^{n}(x_j^{k+1}/x_j^k), \tag{3.7}$$

and update $(D_k)_{ii}$ only if

$$\delta^2(D_k)_{ii}/x_i^{k+1} \notin [1/2, 2]. \tag{3.8}$$

In our problem, for each $x_k(i)$ such that (3.8) holds, we would update S_k, S_k^{-1}, and G_1. In k iterations of the algorithm (where $l_0 = z^*$), we would require $O(kn^{0.5})$ of these updates. In this case, using (3.2) results in a complexity of $O((n^{0.5}n_1^2 + n\bar{n} + n_0^3)nL^2)$ for the entire algorithm, or, if $N \sim n_0 \sim n_1$, $O(n^{2.5}L^2)$, compared to Karmarkar's general result of $O(n^{3.5}L^2)$. For unknown objective values, we could use Karmarkar's canonical form and (together with the results in the next section) obatain a similar reduction.

Several practical problems occur with using rank–one updates of the explicit inverses. First, we cannot guarantee the lower bound estimates in our algorithm unless $l_0 = z^*$. We can, in practice, however, update l_k only after several iterations or use $c^T x_k$ (with different termination condition) in place of l_{k+1} in Step 4 and still achieve low iteration counts (Lustig[20]). The second problem is that we cannot guarantee the same complexity order if we allow the large values of α used in practice (see, e.g., the discussion in [15]). Shanno [22] observed that the complexity can be viewed as a function of α, that no bound on α appears necessary, and that intervals $[1/\rho, \rho]$ for $\rho > 2$ may also be used in (3.8) to increase computational efficiency.

The third problem in Karmarkar's rank–one updating scheme is its use of explicit inverses. Shanno also describes how to avoid this problem by using Fletcher and Powell's [12] updating formula for Cholesky factorisations (with the same complexity order as the inverse updates). This appears to achieve substantial increases in efficiency. In our case, we would use this procedure to update our Cholesky factorisations of S_k, G_1, and G_2, and, thereby, achieve a practical and efficient use of our projection method.

4. Dual Formulation

The dual problem to (1.1) can also be solved by a variant of Karmarkar's algorithm. The program has block-angular structure which leads to computational reductions similar to those of Section 3. An advantage

is also gained for the use of Karmarkar's algorithm in approximation schemes.

The dual of (1.1) is

$$\begin{array}{lll} \max & b_0^T\pi_0 + \sum_{k=1}^N b_k^T\pi_k & \\ \text{subject to} & A_0^T\pi_0 + \sum_{k=1}^N A_k\pi_k \leq c_0 & \\ & \overline{W}_k^T\pi_k \leq c_k, k=1,\ldots,N. & \end{array} \tag{4.1}$$

An alternative in form (2.1) can be easily obtained. Rewrite (4.1) as

$$\begin{array}{lll} \min & -\sum_{k=0}^N \bar{b}_k^T y_k & \\ \text{subject to} & \sum_{k=0}^N \overline{A}_k^T y_k = c_o, & \\ & \overline{W}_k^T y_k = c_k, k=1,\ldots,N; & \\ & y_k \leq 0, k=0,\ldots,N; & \end{array} \tag{4.2}$$

where $\overline{A}_0^T = (A_o^T, -A_0^T, I), \overline{A}_k^T = (A_k^T, -A_k^T, 0)$, $\bar{b}_k^T = (b_k^T, -b_k^T, 0)$, $\overline{W}_k^T = (W_k^T, -W_k^T, I)$, $k=1,\ldots,N$, and the y_k variables are defined accordingly.

The computational effort is again dominated by calculating M^{-1} where $M = \tilde{A}D^2\tilde{A}^T + cc^T$ for $\tilde{A}$ the constraint matrix in (4.2).

Note that

$$\tilde{A}D^2\tilde{A} = \begin{pmatrix} \sum_{k=0}^N \overline{A}_k^T D_k^2 \overline{A}_k & \overline{A}_1^T D_1^2 \overline{W}_1 & \cdots & \overline{A}_N^T D_N^2 \overline{W}_N \\ \overline{W}_1^T D_1^2 \overline{A}_1 & \overline{W}_1^T D_1^2 \overline{W}_1 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ \overline{W}_N^T D_N^2 \overline{A}_N & 0 & \cdots & \overline{W}_N^T D_N^2 \overline{W}_N \end{pmatrix}.$$

Now observe that $M = P + UV^T$ where

$$P = \begin{pmatrix} \sum_{k=0}^N \overline{A}_k^T D_k^2 \overline{A}_k & 0 & \cdots & 0 \\ \overline{W}_1^T D_1^2 \overline{A}_1 & \overline{W}_1^T D_1^2 \overline{W}_1 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ \overline{W}_N^T D_N^2 \overline{A}_N & 0 & \cdots & \overline{W}_N^T D_N^2 \overline{W}_N \end{pmatrix},$$

$$U = \begin{pmatrix} I & c_0 \\ 0 & c_1 \\ \vdots & \vdots \\ 0 & c_N \end{pmatrix},$$

and

$$V = \begin{pmatrix} 0 & c_0 \\ \overline{W}_1 D_1^2 \overline{A}_1 & c_1 \\ \vdots & \vdots \\ \overline{W}_1^T D_N^2 \overline{A}_N & c_N \end{pmatrix}.$$

The Sherman-Morrison-Woodbury formula yields

$$M^{-1} = P^{-1} - P^{-1}U(I + V^T P^{-1} U)^{-1} V^T P^{-1}.$$

Let

$$H = \sum_{k=0}^{N} \overline{A}_k^T D_k^2 \overline{A}_k,$$

$$H_k = \overline{W}_k^T D_k^2 \overline{A}_k, k = 1, \ldots, N,$$

and

$$J_k = \overline{W}_k^T D_k^2 \overline{W}_k, k = 1, \ldots, N,$$

then

$$P^{-1} = \begin{pmatrix} H^{-1} & 0 & \cdots & 0 \\ -J_1^{-1} H_1 H^{-1} & J_1^{-1} & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ -J_N^{-1} H_N H^{-1} & 0 & \cdots & J_N^{-1} \end{pmatrix} \tag{4.4}$$

and

$$(I + V^T P^{-1} U) = \begin{pmatrix} I - \sum_{k=1}^{N} H_k^T J_k^{-1} H_k H^{-1} & -\sum_{k=1}^{N} (H_k^T J_k^{-1} H_k H^{-1} c_0 - H_k^T J_k^{-1} c_0) \\ c_o^T H^{-1} - \sum_{k=1}^{N} c_k^T J_k^{-1} H_k H^{-1} & I + c_0^T H^{-1} c_0 + \sum_{k=1}^{N} (c_k^T J_k^{-1} H_k H^{-1} c_0 + c_k^T J_k^{-1} c_k) \end{pmatrix}. \tag{4.5}$$

Equations (4.3), (4.4), and (4.5) can then be used to develop another projection method for (4.2).

Theorem 4.2. The overall running time of the projection using (4.3–5) is $O(N(n_1^3 + n_1^2 n_0 + n_1 n_0^2))$.

Proof: The computational effort in each step is:

work	number of operations
Calculate H^{-1}	$O(Nn_1n_0^2)$
Calculate $J_k^{-1}, k = 1 \ldots, N$	$O(Nn_1^3)$
Calculate $H_k H^{-1}, k = 1, \ldots, N$	$O(Nn_1n_0^2)$
Calculate $H_k^T J_k^{-1}, k = 1, \ldots, N$	$O(Nn_1^2n_0)$
Calculate $(I + V^T P^{-1} U)^{-1}$	$O(n_0^3)$
Multiply A with a vector	$O(mn_0 + Nn_1(m_1 + n_1))$
Multiply P^{-1} with a vector	$O(n\overline{n})$
Multiply V^T with a vector	$O(n_0 n)$

All other effort is dominated by these operations. The number of operations is $O(N(n_1^3 + n_1^2 n_0 + n_1 n_0^2))$, and the result is obtained as in Theorem 3.2.■

Theorem 4.2 gives the same complexity as Theorem 3.2, so, again for $N \sim n_0 \sim n_1$, we have $O(n^3 L^2)$ complexity for the full algorithm. We can again use Cholesky factorizations to make the projection with (4.3–5) practical.

5. Dual Use in Approximations

The dual provides an additional benefit. In using (3.1) to approximate a stochastic linear program with more than N realizations of the random vector b_k, additional realizations $b_{N+1}, \ldots, b_{N+l}$ are added to (3.1) and the probabilities $p_k, k = 1, \ldots, N+l$ are updated. (See Birge and Wets [6] for procedures to do this.) A feasible solution to the new problem (3.1) with $N' = N + l$ is not readily found but given a feasible solution (4.2) another can be found easily if $T^1 = T^2 = \cdots = T^N = T$ and $q^1 = q^2 = \cdots = q^N = q$. Under these assumptions let the old solution be $(y_0^{\text{old}}, \ldots, y_N^{\text{old}})$ where

$$\overline{W}^T y_k^{\text{old}} = p_k q, k = 1, \ldots, N,$$

and

$$\overline{A}_o^T y_0^{\text{old}} + \sum_{k=1}^{N} \overline{T}^T y_k^{\text{old}} = c_0,$$

where $\overline{T} = (T, -T, 0)$.

Define a new solution by

$$\begin{aligned} y_0^{\text{new}} &= y_0^{\text{old}}, \\ y^{\text{new}} &= (\sum_{i=1}^{N} y_i^{\text{old}}) p_k^{\text{new}}, k = 1, \ldots, N+l. \end{aligned} \tag{5.1}$$

The solution in (5.1) is then feasible in (4.2) with objective value $b_0^T y_0^{\text{old}} + \sum_{k=1}^{N+l} b_k^T (\sum_{i=1}^{N} y_i^{\text{old}}) p_k^{\text{new}}$. The total effort for solving the new (4.2) should be less than resolving (3.1) because no procedures to find a feasible interior point are necessary (assuming, of course, that y^{old} is interior). The objective is generally close to the old optimal value of (4.2) and should be close to the new optimal value. The effort factor for the number of iterations to solve the new (4.2) should therefore be much smaller than $O(nL)$.

A similar procedure can be used in the primal problem (3.1) if the randomness is limited to the objective function, i.e., $T^1 = T^2 = \cdots = T^N = T$ and $h^1 = \ldots = h^N = h$ but $q^i \neq q^j$ for all $i \neq j$. Again,a starting

feasible solution can be easily found by letting

$$x_k^{\text{new}} = x_k^{\text{old}}, k = 0, \ldots, N,$$
$$x_j^{\text{new}} = \text{argmin}_{x_i^{\text{old}}}\{q^j x_i^{\text{old}}, i = 1, \ldots, N\}, j = N+1, \ldots, N+l. \tag{5.2}$$

Now let $\bar{p} = \sum_{N+1}^{N+l} p_j^{\text{new}}$, z_{old} be the objective of the old (3.1) with x_k^{old} and let

$$z_{\text{new}}^0 = \bar{p} c^T x_0^{\text{new}} + \sum_{j=N+1}^{N+l} p_j^{\text{new}} q^j x_j^{\text{new}}.$$

The new initial objective value in (3.1) is then

$$z_{\text{new}} = (1-\bar{p}) c^T x_0^{\text{old}} + \sum_{j=1}^{N} p_j^{\text{new}} q^j x_j^{\text{old}} + z_{\text{new}}^0,$$

and we wish to find

$$z_{\text{new}}^* = \min\{(1-\bar{p}) c^T x_0^{\text{old}} + \sum_{j=1}^{N} p_j^{\text{new}} q^j x_j + z'_N\}, \tag{5.3}$$

where $z'_N = \bar{p} c^T x_0 + \sum_{j=N+1}^{N+l} p_j^{\text{new}} q^j x_j$. Let z^* be the optimal value of the old (3.1) and let x^{old} be an interior solution of the old (3.1) such that $z_{\text{old}} < z^* + \epsilon$. Now assume that $p_j^{\text{new}}/(1-\bar{p})$, i.e. that the old probabilities have the same relative values in the new problem, so that $(1-\bar{p})z^*$ is the minimum of $(1-\bar{p}) c^T x_0 + \sum_{j=1}^{N} p_j^{\text{new}} q^j x_j$. We then have

$$z_{\text{new}}^* \leq (1-\bar{p}) z_{\text{old}} + \max z'_N$$
$$z_{\text{new}}^* \geq (1-\bar{p})(z_{\text{old}} - \epsilon) + \min z'_N. \tag{5.4}$$

Let L' be the size of the data for the objective and constraints relating only to z'_N. This yields the following result.

Theorem 5.1 The overall running time of Algorithm 3.1 starting from a solution defined by (5.2) for $N+l$ realisations of the objective q^k, assuming $T^1 = \cdots = T^{N+l} = T$, $h^1 = \cdots = n^{N+l} = h$, $p_j^{\text{new}}/(1-\bar{p}) = p_j^{\text{old}}$, and $\epsilon < 2^{L'}$, is $O((N+l)(n_1^3 + n_1^2 n_0 + n_1 n_0^2) n L L')$.

Proof: The number of operations per iteration follows from Theorem 3.2. The number of iterations $O(nL')$ follows from (5.4) for $2^{O(L')} \geq z'_N \geq -2^{O(L')}$.

Theorem 5.1 provides a worst–case bound that can be used in solving successively larger stochastic linear programs. The entire approximating process can then be seen as a variable dimension variant of

Karmarkar's algorithm in which the number of iterations is $O(\bar{l}nL')$ where $\bar{l}$ additional problems of size L' are added after the initial solution of (3.1).

This result contrasts with the alternative (simplex-based) approaches to stochastic linear programs, for which, the complexity of solving each new problem depends on the size of the entire problem. Good bounds on x'_N that would often be available in practice could further reduce the computational effort.

For a single SLPF, we can also compare the projective algorithm with simplex- based methods if we assume average cases for iteration counts. The best available results for stochastic linear programming appear to be in uses of the L-shaped method of Van Slyke and Wets (Birge [2]). It appears that the running time is typically then $O(N\bar{n}^3) = O(n^2)$ (Birge and Louveaux [5]). If we use rank-one updates in the projective method and assume an average $O(\log n)$ iterations as has been conjectured (Karmarkar [18]), then the projective algorithm has running time $O(n^{1.5}\log n)$ (ignoring L which is assumed in both estimates). This observation combined with the potential for sequential approximation indicates the promise for the projective algorithm in stochastic linear programming. Further computational experience will be required to bear this out.

6. Special Cases

The work of the variants of Karmarkar's algorithm is reduced for the following special cases.

(1) **Simple recourse** [28]

In this case, $W = (I, -I)$, where I is the unit matrix in $\Re^{m_1 \times m_1}$, and $n_1 = 2m_1$. Write $x_k = (x_k^+, x_k^-), x_k^+, x_k^- \in \Re^{m_1}$, for $k = 1, \ldots, N$ in (3.1). Let $D_k = \text{diag}\{D_k^+, D_k^-\}$ in (3.2), where $D_k^+, D_k^- \in \Re^{m_1 \times m_1}$, for $k = 1, \ldots, N$. Then in Theorem 3.1, we have

$$\begin{aligned} S_k &= W D_k^2 W^T = (D_k^+)^2 + (D_k^-)^2, \\ S_k^{-1} &= \text{diag}\{((a_k^+)_1^2 + (a_k^-)_1^2)^{-1}, \ldots, ((a_k^+)_{m_1}^2 + (a_k^-)_{m_1}^2)^{-1}\}, \end{aligned} \tag{6.1}$$

where $D_k^+ = \text{diag}\{(a_k^+)_1, \ldots, (a_k^+)_{m_1}\}$ and $D_k^- = \text{diag}\{(a_k^-)_1, \ldots, (a_k^-)_{m_1}\}$, for $k = 1, \ldots, N$. Therefore, S^{-1} can be easily calculated in (3.2).

The work is reduced but the order of the algorithm is the same. Similar modifications can be made in

the dual approach.

(2) **Network recourse** [26]

In this case,

$$W = \begin{pmatrix} E & 0 \\ I & I \end{pmatrix},$$

where E is the network node-arc incidence matrix. The calculation work of S_k and S_k^{-1} can still be reduced but again the order is the same.

References

[1] K. Anstreicher,"Analysis of Karmarkar's algorithm for fractional programming",Yale School of Organisation and Management, Technical Report, Yale University (New Haven, CT, 1985).

[2] J. Birge,"Decomposition and partitioning methods for multistage stochastic linear programs", **Operations Research** 33 (1985), 989–1007.

[3] J. Birge,"A Dantzig-Wolfe decomposition variant equivalent to basis factorisation", **Mathematical Programming Study**, 24 (1985), 43-64.

[4] J. Birge,"The relationship between the L-shaped method and dual basis factorisation for stochastic linear programming", in Y. Ermoliev and R. Wets, eds., **Numerical Methods in Stochastic Programming**, to appear.

[5] J. Birge and F. Louveaux, "A multicut algorithm for two–stage stochastic linear programs", Department of Industrial and Operations Engineering Technical Report 85-15, University of Michigan (Ann Arbor, MI, 1985).

[6] J. Birge and R. Wets," Designing approximation schemes for stochastic optimisation problems, in particular, for stochastic programs with recourse", **Mathematical Programming Study** 27 (1986) 54–102.

[7] J. Bisschop and A. Meeraus,"Matrix augmentation and partitioning in the updating of the basis inverse", **Mathematical Programming**, 13 (1977) , 241-254.

[8] J. Bisschop and A. Meeraus,"Matrix augmentation and structure preservation in linearly constrained control problems", **Mathematical Programming**, 18 (1980), 7-15.

[9] A. Charnes, T. Song, and M. Wolfe,"An explicit solution sequence and convergence of Karmarkar's algorithm", Research Report CCS501, Center for Cybernetic Studies, University of Texas at Austin, (Austin, TX,1984).

[10] G. Dantsig, "Upper bounds, secondary constraints and block triangularity in linear programming",**Econometrica** 23 (1955) 174–183.

[11] G. Dantsig and A. Mandansky,"On the solution of two-stage linear programs under uncertainty", **Proc. Fourth Berkeley Symposium on Mathematical and Probability**, Vol. 1, University of California Press, Berkeley, 1961, pp. 165- 176.

[12] R. Fletcher and M. Powell,"On the modification of LDL^T factorisations", **Math. Comp.** 28 (1974) 1067–1087.

[13] R. Fourer,"Staircase matrices and systems", **SIAM Review**, 26 (1984) , 1-70.

[14] D. Gay,"A variant of Karmarkar's linear programming algorithm for problems in standard form", Numerical Analysis Manuscript 85–10, A T and T Bell Laboratories (Murray Hill,NJ,1985), to appear in **Mathematical Programming**.

[15] P. Gill, W. Murray, M. Saunders, J. Tomlin and M. Wright,"On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method", Systems Optimization Laboratory Technical Report SOL 85–11, Stanford University (Stanford, CA, 1985).

[16] G. Golub and C. Van Loan,**Matrix Computations**, John Hopkins University Press, Baltimore, 1983.

[17] P. Kall,"Computational methods for solving two-stage stochastic linear programming problems", **Z. Angew. Math. Phys.**, 30 (1979), 261-271.

[18] N. Karmarkar, "A new polynomial–time algorithm for linear programming", presented at the ORSA/TIMS Joint National Meeting, Dallas, Texas, 1984.

[19] N. Karmarkar,"A new polynomial-time algorithm for linear programming", **Combinatorica**, 4(1984), 373–395.

[20] I. Lustig,"A practical approach to Karmarkar's algorithm", Systems Optimisation Laboratory Technical Report SOL 85-5, Stanford University (Stanford, CA, 1985).

[21] A. Perold and G. Dantzig,"A basis factorization method for block triangular linear programs" in: **Sparse Matrix Proceedings** , 1978, I. Duff and G. Stewart, eds., SIAM Publications, Philadelphia, pp. 283-312.

[22] D. Shanno,"Computing Karmarkar projections quickly", Graduate School of Administration Working Paper 85–10, University of California, Davis (Davis,CA,1985).

[23] B. Strasicky,"Some results concerning an algorithm for the discrete recourse problem", in: **Stochastic Programming** , M. Dempster, ed., Academic Press, London, 1980, pp. 263-274. [24] M. Todd and B. Burrell,"An extension of Karmarkar's algorithm for linear programming using dual variables", School of Operations Research and Industrial Engineering Technical Report No. 648, Cornell University, (Ithaca, NY, 1985).

[25] R. Van Slyke and R. Wets,"L-shaped linear programs with applications to optimal control and stochastic programming", **SIAM J. Appl. Math.** 17 (1969) 638-663.

[26] S. Wallace,"On network structured stochastic optimisation problem", Report No. 842555-8, Chr. Michelsen Institute, (Bergen, Norway,1985).

[27] R. Wets,"Stochastic programming: Solution techniques and approximation schemes", in: **Mathematical Programming: The State of the Art , 1982**, A. Bachem, M. Groetschel and B. Korte, eds., Springer-Verlag, Berlin, 1983, pp. 566-603.

[28] R. Wets,"Large scale linear programming techniques in stochastic programming", in: Y. Ermoliev and R. Wets, eds., **Numerical Methods in Stochastic Programming**, to appear.

[29] C. Winkler,"Basis factorisation for block-angular linear programs : unified theory of partition and decomposition using the simplex method", Systems Optimisation Loaboratory Technical Report SOL 74-19 (Stanford, CA, 1974).

[30] Y. Ye,"Cutting-objective and scaling methods - a 'simple' polynomial algorithm for linear programming", Engineering-Economic Systems Department Technical Report, Stanford University (Stanford, CA, 1985).

END

12-87

DTIC